虚拟存储器和CACHE异同

- **相同之处**
  - 目的为了提高系统性能
  - 数据分为小信息块，作为基本的传递单位
  - 都存在地址映射，替换算法，更新策略
  - 按照局部性原理，将活跃的数据放到高速部件中
- **不同之处**
  - Cache解决系统速度问题　苏尼存储器解决主存容量问题
  - Cache 全部由硬件实现，是硬件存储器 虚拟存储器有OS和硬件共同实现，是逻辑上的存储器
  - Cache对所有程序员透明，虚拟存储器对应用程序员透明，对系统程序员不透明
  - 虚拟存储器不命中对系统性能影响更大
  - CPU只能与Cache和主存直接交互　虚拟存储系统只能先将数据从硬盘调入主存，不能与CPU直接通信


3.1 ~3.2存储器的层次结构
3.1.1存储器分类
层次分类
主存
存放计算机运行时的程序与数据
可以被CPU直接访问，也可以与Cache和辅存进行交换数据
容量小，速度快，价格高
辅存
存储当前暂时不用的程序和数据，以及一些永久保留数据
不可以与CPU直接交互
容量大，速度慢，价格低
高速缓冲存储器（cache）
位于主存与CPU之间，缓解CPU与主存之间速度差距问题
CPU可以直接访问
容量小，速度与CPU相匹配，价格高 高档计算机将他们放在CPU中
存储介质分类
磁表面存储器：磁盘 磁带
磁芯存储器半导体存储器（MOS型存储器 双极型存储器）
光存储器：光盘
存取方式分类
随机存储器（RAM）
随机存取，读写方便，使用灵活
用途：主存 高速缓冲存储器
类型： 静态RAM(触发器原理） 动态RAM（电容充电原理）
只读存储器（ROM）
只能读出不能写入
用途： 与随机存储器构成主存 存储固定不变的程序（甚至操作系统的固化）
ROM也派生出可以反复重写的类型
rom与ram都是随机存取，广义上的只读存储器可以通过电擦除写入的，写入速度比读速度慢
信息可保存性
易失性存储器
RAM
断电后，仍然可以保持信息
非易失性存储器存储器
ROM、磁表面存储器、光存储器
读出类型
破坏性读出：读出后需要对存储信息再生
非破坏性读出：不会对原存储信息破坏
3.1.2 存储器的性能指标
计算机追求目标：大容量，低成本，高速度（联系自身买电脑）
存储容量=存储字数*字长
单位成本=总成本/总容量
存储速度： 数据传输率 = 数据宽度/存储周期
存取时间：存储器启动到完成操作，分为读出时间与写入时间
存取周期（读写周期、访问周期）：存储器进行一次完整读写操作需要时间
主存带宽：数据传输率，每秒主存进出信息的最大数量
存储周期一般大于存取时间（读写操作后需要内部复原时间）
3.2 存储器的层次化结构
存储系统目标：更大更快更便宜
Cache-----主存层次 解决CPU和主存速度不匹配问题
主存-------辅存层次 解决存储系统容量问题
主存----辅存层次不断发展，衍生出虚拟存储器系统

3.3 半导体随机存储器（上）

3.3.1 半导体存储芯片

- 存储矩阵：大量相同的位存储单元阵列构成
- 译码驱动：地址信号翻译成对应存储单元的选通信号
- 读写电路：完成读写操作
- 读/写控制线：决定芯片是读还是写
- 片选线：确定那个芯片被选中
- 地址线：单项输入，位数与存储字的个数有关
- 数据线：双向的，位数与读出或写入的数据位数有关

数据线数与地址线数共同反映存储芯片容量大小

- 半导体随机存储器分类（存储原理不同）
  - SRAM：高速缓存
  - DRAM：主存
- 74138译码器

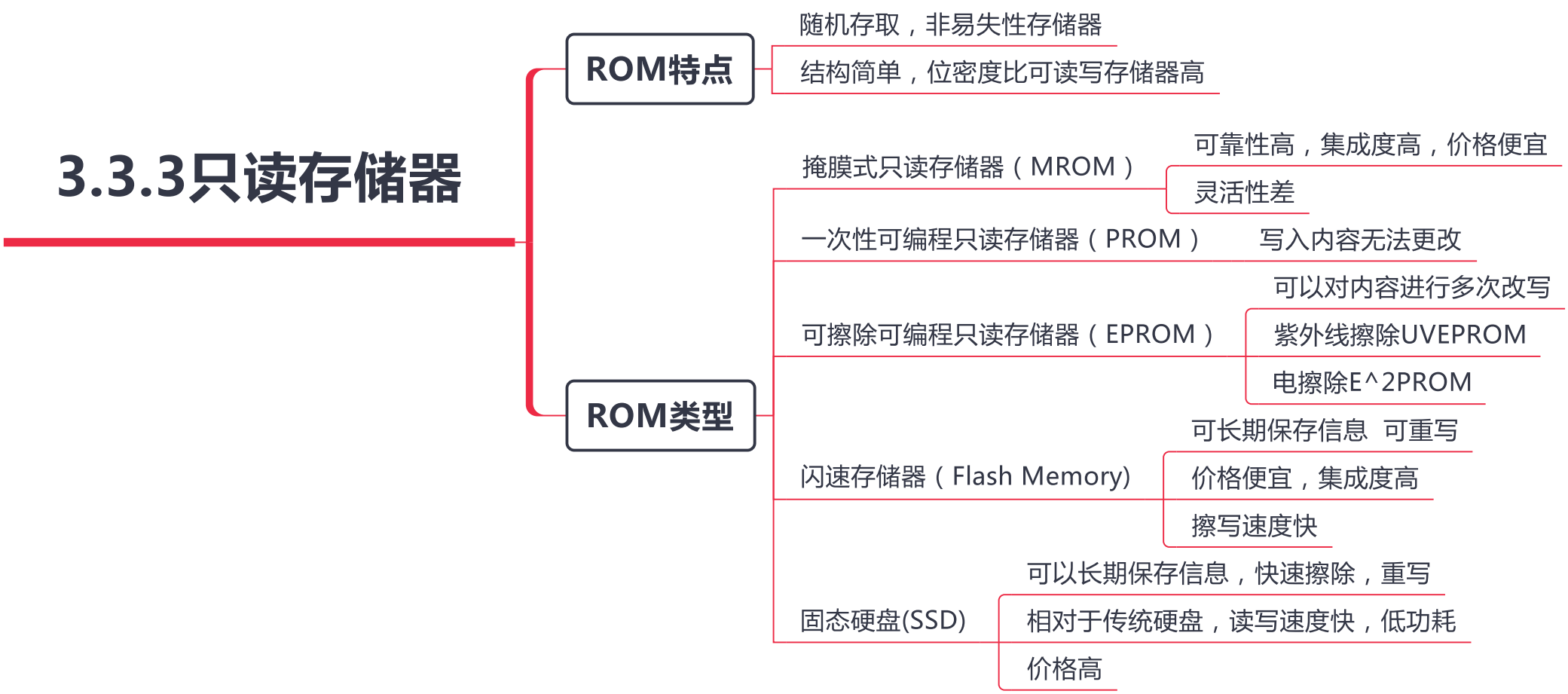
3.3.2 SRAM和DRAM

- SRAM
  - 使用双稳态触发器（六管MOS）记忆信息
  - 非破坏性读出，易失性存储器
  - 存取速度快，集成度低，功耗大，成本高,常用来组成高速缓冲存储器
  - 同时送行列地址
- DRAM
  - 利用电荷存储信息
  - 破坏性读出，易失性存储器
  - 存取速度慢，集成度高，功耗低，容量大 ，成本低， 常用来组成主存系统
  - 分两次送行列地址
  - 刷新方法（DRAM特有）
    - 刷新单位是行
    - 集中刷新
      - 刷新时间固定 存在死区
      - 读写不受刷新影响，存取速度快
      - 死区不能访问存储器
    - 分散刷新
      - 将每行刷新分散到各个工作周期中
      - 没有死区
      - 存取周期边长，降低整机速度
    - 异步刷新
      - 集中刷新和异步刷新结合
      - 缩短了死时间，又提高了整机速度
    - 透明刷新
      - 刷新安排在译码阶段，不存在死时间

死区：在刷新的时候，停止对存储器的读写操作，称为死时间

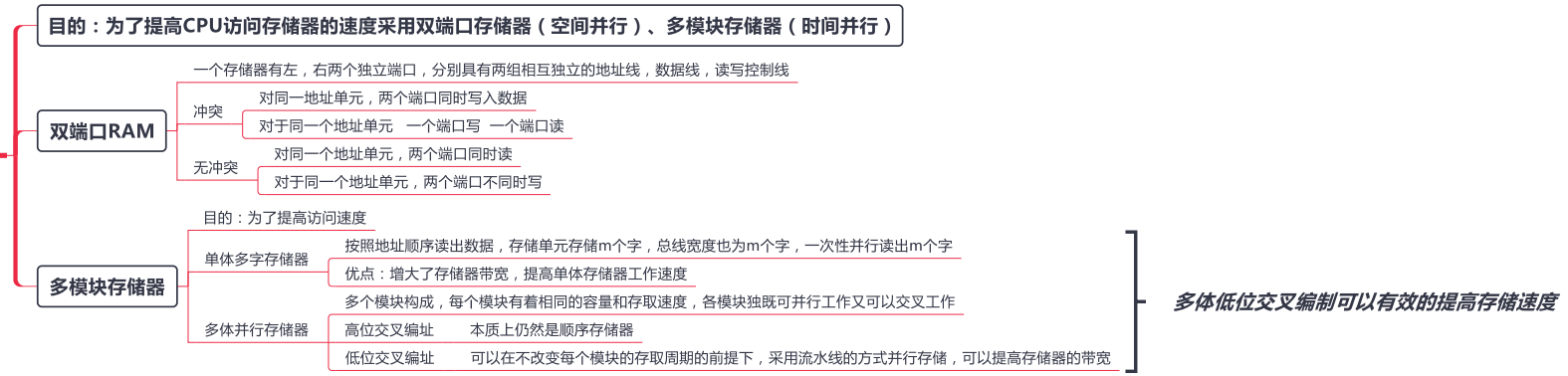
- 存储器的读写周期
  - RAM读周期：存储芯片进行两次连续读操作时，必须间隔的时间，读周期总是大于等于读出时间
  - RAM写周期：数据总线上的信息能够可靠的写入存储器

3.3.3只读存储器


ROM特点
随机存取，非易失性存储器
结构简单，位密度比可读写存储器高
ROM类型
掩膜式只读存储器（MROM）
可靠性高，集成度高，价格便宜
灵活性差
一次性可编程只读存储器（PROM）
写入内容无法更改
可擦除可编程只读存储器（EPROM）
可以对内容进行多次改写
紫外线擦除UVEPROM
电擦除E^2PROM
闪速存储器（Flash Memory)
可长期保存信息 可重写
价格便宜，集成度高
擦写速度快
固态硬盘(SSD)
可以长期保存信息，快速擦除，重写
相对于传统硬盘，读写速度快，低功耗
价格高

3.4 主存储器和CPU的连接

3.5 双端口RAM和多模块存储器



3.6高速缓冲存储器

程序访问的局部性原理

- 时间局部性：将要使用的信息，可能是现在正在使用的信息
- 空间局部性：将来使用的信息，可能在正在使用信息的存储空间的附近
- 基于局部性原理创造出高速缓冲技术（Cache）

CACHE基本工作原理

- 通常使用SRAM制造
- 存储主存中最为活跃的信息副本，按照某种策略将这些活跃的信息存入到Cache中
- CPU发出读请求
  - Cache命中：直接对Cache进行读操作
  - Cache不命中：CPU访问主存操作，并且将访问数据送入到Cache中

cpu与cache之间交换数据的基本单位是 字
cache与主存之间交换数据的基本单位是 cache块
注意：某些计算机也可能是同时访问cache和主存

CACHE与主存的映射方式

- 按照一定的规则将主存中的某些数据存入到Cache中
- 直接映射
  - 主存数据块只能装入Cache中的唯一位置
  - 地址结构　主存字块标记　Cache字块地址　字块内地址

冲突率高，利用率低，实现简单

- 全相联映射
  - 主存数据库可以放在Cache中的任何位置
  - 地址结构　主存字块标记　字块内地址

地址变换慢，实现成本高　比较灵活 冲突率低

- 组相邻映射
  - 将Cache分为不同的组，主存的数据库可以装入一组内的任何位置
  - 地址结构　主存字块标记　组地址　字块内地址

CACHE中的替换算法

- 随机算法：随机确定替换的Cache块
  - 优点：实现简单
  - 缺点：没有依据局部性原理，命中率低
- 先进先出算法（FIFO）：最早调入的行进行替换
  - 优点：容易实现
  - 缺点:没有依据局部性原理
- 近期最少使用算法（LRU）:根据局部性原理，选择近期内最久没有访问的存储行
  - 优点：平均命中率高的
  - 缺点：需要设置计数器比较存储行

堆栈类算法

- 最不经常使用算法：一段时间内访问次数最少的存储行换出

CACHE写策略

- 写命中
  - 全写法（写直通法、write-through）:对Cache写命中后，数据同时写入Cache 和 主存
    - 实现简单，随时保持主存数据正确性
    - 增加了访存次数，降低了效率
  - 写回法（write-back）：对Cache命中时，只修改Cache内容，不立即写入主存
    - 减少了访存次数
    - 存在数据不一致的隐患，同时需要设置一个脏位
- 写不命中
  - 写分配法（write-allocate):加载主存中的块到Cache中，然后更新Cache块
    - 试图使用空间局部性原理
    - 每次不命中都要从主存中调块
  - 非写分配法（not-write-allocate):只写入主存，不进行调块
- 多级Cache（通常为3级）　可以有效避免频繁写时造成的写缓冲饱和和溢出

非写分配法与全写法搭配
写分配法与写回法搭配

3.7虚拟存储器

基本概念

- 将主存或者辅存的地址空间统一编址
- 实地址对应的是主存地址空间
- 使用虚地址需要辅助硬件找出虚地址和实地址之间的关系，并对其对应存储单元装入状态进行判断
- 实际情况：需要使用的先送入主存，暂时不用的放在磁盘中

页式虚拟存储器

- 以页为基本单位的虚拟存储器称为页式虚拟存储器
- 计算过程
  - 虚拟地址 = 虚页号+页内地址
  - 虚页号+页表起始地址 = 页表地址　　根据页表地址查找实页号
  - 实页号+页内地址 = 实地址
  - 子主题 4

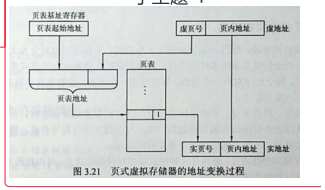


图 3.21　页式虚拟存储器的地址变换过程

优点：页面长度固定 调入方便
缺点：零头浪费 对页的处理、保护、共享不是特别方便

段式虚拟存储器

- 按照程序的逻辑结构划分
- 计算过程
  - 虚地址 = 段号 + 段内地址
  - 段号+段表起始地址 = 段表地址　　查询段表数据
  - 实地址 = 段表所得数据+ 段内地址

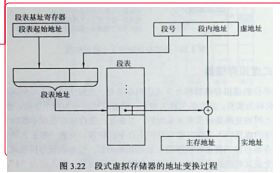


图 3.22　段式虚拟存储器的地址变换过程

优点： 段分界与程序分界相对应 易于编译、管理、修改、保护、共享
缺点：段长分配不便 存在段间碎片

段页式存储器

- 先将程序按逻辑分段，再将每段分为固定大小页
- 段长必须是页的整数倍
- 虚地址 = 段号 + 段内页号 + 页内地址

优点： 可以按段实现共享和保护，同时也有着页的调用方便
缺点：地址变换要两次查表，开销较大

快表（TLB）

- 根据局部性原理，将一些经常访问的页放入高速缓冲器中构成快表，可以极大提高查询的效率
- 采用相联存储器构成，可以按照内容查询
- 访问顺序：TLB--->页表--->Cache--->主存
- 命中情况
  - Cache命中，page必然命中，TLB不一定命中
  - Cache不命中，无法退出TLB与page命中情况
  - Page不命中，Cache和主存不会命中，此时要执行调页策略

只要抓住存储数据的来源就可以推导出命中情况